

# Semi-Supervised Nonlinear Distance Metric Learning via Forests of Max-Margin Cluster Hierarchies

David M. Johnson  
davidjoh@buffalo.edu

Caiming Xiong  
cxiong@buffalo.edu

Jason J. Corso  
jcorso@buffalo.edu \*

## Abstract

Metric learning is a key problem for many data mining and machine learning applications, and has long been dominated by Mahalanobis methods. Recent advances in nonlinear metric learning have demonstrated the potential power of non-Mahalanobis distance functions, particularly tree-based functions. We propose a novel nonlinear metric learning method that uses an iterative, hierarchical variant of semi-supervised max-margin clustering to construct a forest of cluster hierarchies, where each individual hierarchy can be interpreted as a weak metric over the data. By introducing randomness during hierarchy training and combining the output of many of the resulting semi-random weak hierarchy metrics, we can obtain a powerful and robust nonlinear metric model. This method has two primary contributions: first, it is semi-supervised, incorporating information from both constrained and unconstrained points. Second, we take a relaxed approach to constraint satisfaction, allowing the method to satisfy different subsets of the constraints at different levels of the hierarchy rather than attempting to simultaneously satisfy all of them. This leads to a more robust learning algorithm. We compare our method to a number of state-of-the-art benchmarks on  $k$ -nearest neighbor classification, large-scale image retrieval and semi-supervised clustering problems, and find that our algorithm yields results comparable or superior to the state-of-the-art, and is significantly more robust to noise.

**Index terms**— Metric learning, nonlinear, semi-supervised, max-margin clustering, hierarchical clustering

## 1 Introduction

Many elemental data mining problems—nearest neighbor classification, retrieval, clustering—are at their core dependent on the availability of an effective means of measuring pairwise distance. Ad hoc selection of a metric, whether by relying on a standard such as Euclidean distance or attempting to select a domain-appropriate kernel, is unreliable and inflexible. We thus approach metric selection as a learning problem, and attempt to train strong problem-specific distance measures using data and semantic information.

A wide range of methods have been proposed to address this learning problem, but the field has traditionally been dominated by algorithms that assume a linear model of distance, particularly Mahalanobis metrics [1]. Linear methods have primarily benefited from two advantages. First, they are generally easier to optimize, allowing for faster learning and in many cases a globally optimal solution to the proposed problem [2, 3, 4, 5]. Second, they allow the original data to be easily projected into the new metric space, meaning the metric can be used in conjunction with other methods that operate only on an explicit feature representation (most notably approximate nearest neighbor methods—needed if the metric is to be applied efficiently to large-scale problems).

However, for many types of data a linear approach is not appropriate. Images, videos, documents and histogram representations of all kinds are ill-suited to linear models. Even an ideal Mahalanobis metric will be unable to capture the true semantic structure of these types of data, particularly over larger distances where local linearity breaks down. Kernelized versions of popular Mahalanobis methods [2, 6] have been proposed to handle such data, but these approaches have been limited by high complexity costs. For this reason, researchers have begun to seek alternate metric models that are inherently capable of handling nonlinear data.

These nonlinear metrics are necessarily a broad class of models, encompassing a range of learning modalities and metric structures. One early example of nonlinear metrics (for facial recognition, in this case) by Chopra et al. [7] was based on deep learning strategies. The method was effective, but required long training times and extensive tuning of hyperparameters. Other methods sought to resolve the problem by taking advantage of local linearity in the data, and learning multiple localized linear metrics [8, 9, 10, 11]. These techniques have generally proven superior to single-metric methods, but have also tended to be expensive.

Most recently, several works have explored metrics that take advantage of tree structures to produce flexible nonlinear transformations of the data. Kedem et al. [12] proposed a method that trained a set of gradient-boosted regression trees and added the regression outputs of each region directly to the data, producing an explicit nonlinear transformation that shifted similar points together and dissimilar points apart. However, this method relies on performing regression against a  $d$ -dimensional gradient vector, and may

\*Authors are with the Department of Computer Science and Engineering, SUNY at Buffalo, Buffalo, NY, 14260-2500

thus be prone to overfitting when  $d$  is large relative to the number of training samples.

Finally, our previous work in this area [13] formulated the pairwise-constrained metric learning problem as a pair-classification problem, and solved it by direct application of random forests, yielding an implicit nonlinear transformation of the data. However, while this metric could be trained efficiently, it suffered from poor scalability at inference time due to the lack of an explicit feature representation, which made common metric tasks such as nearest neighbor search expensive on larger datasets.

In order to overcome the limitations of these methods, we propose a novel tree-based nonlinear metric with several advantages over existing algorithms. Our metric first constructs a model of the data by computing a forest of semi-random cluster hierarchies, where each tree is generated by iteratively applying a partially-randomized binary semi-supervised max-margin clustering objective. As a result, each tree directly encodes a particular model of the data’s full semantic structure, and the structure of the tree itself can thus be interpreted as a weak metric. By merging the output from a forest of these weak metrics, we can produce a final metric model that is powerful, flexible, and resistant to overtraining (due to the independent and semi-random nature of the hierarchy construction).

This methodology provides two significant contributions: first, unlike previous tree-based nonlinear metrics, it is semi-supervised, and can incorporate information from both constrained and unconstrained points into the learning algorithm. This is an important advantage in many problem settings, particularly when scaling to larger datasets where only a tiny proportion of the full pairwise constraint set can realistically be collected or used in training.

Second, the iterative, hierarchical nature of our training process allows us to relax the constraint satisfaction problem. Rather than attempting to satisfy every available constraint simultaneously, at each hierarchy node we can optimize an appropriate constraint subset to focus on, leaving others to be addressed lower in the tree (or in other hierarchies in the forest). By selecting constraints in this way, we can avoid situations where we are attempting to satisfy incoherent constraints [14], and thereby better model hierarchical data structures. We can also obtain an algorithm that is more robust to noisy data (see experiments in Section 6.4).

Additionally, we propose a scalable and highly accurate algorithm for obtaining approximate nearest neighbors within our learned metric’s space. This renders the metric tractable for large-scale retrieval or nearest-neighbor classification problems, and overcomes a major limitation our previous tree-based metric.

The remainder of this paper is organized as follows: in Section 2, we describe in detail our formulation for hierarchy-forest-based metric learning. In Section 3 we discuss max-margin clustering and describe our formulation of semi-supervised max-margin clustering for hierarchy learning. In Section 4 we describe our method for fast approximate in-metric nearest neighbor retrieval. In Section 5 we provide a complexity analysis of our method, and in Section 6 we show experimental results in which our

method compares favorably to the state-of-the-art in metric learning.

## 2 Semi-supervised max-margin hierarchy forests

In this section we describe in detail our Hierarchy Forest Distance (HFD) model, as well as our procedures for training and inference. The structure of the HFD model draws some basic elements from random forests [15], in that it is composed of  $T$  trees trained independently in a semi-random fashion, with individual nodes in the trees defined by a splitting function that divides the local space into two or more segments. Each hierarchy tree represents a distance function  $\mathcal{H}(a, b)$ , and the overall distance function is

$$D(a, b) = \frac{1}{T} \sum_{t=1}^T \mathcal{H}_t(a, b) . \quad (1)$$

However, HFD is conceptually distinct from random forests (and the Random Forest Distance (RFD) metric [13]) in that the individual components of the forest represent cluster hierarchies rather than decision trees. We discuss this distinction and its implications in Section 2.2.

### 2.1 Hierarchy forest distance

The full hierarchy forest distance is effectively the mean of a number of weak distance functions  $\mathcal{H}_t$ , each corresponding to one hierarchy in the forest. These distance functions, in turn, are representations of the structure of the individual hierarchies—the further apart two instances fall within a hierarchy, the greater the distance between them. Specifically, we formulate each metric as a modified form of the hierarchy distance function we previously proposed for use in hierarchy comparison [16]:

$$\mathcal{H}_t(a, b) = \begin{cases} 0 & \text{if } \mathcal{H}_{tl(a,b)} \text{ is a leaf node} \\ p_t(a, b) \cdot \frac{|\mathcal{H}_{tl(a,b)}|}{N} & \text{otherwise,} \end{cases} \quad (2)$$

where  $\mathcal{H}_t$  represents a particular hierarchy,  $a$  and  $b$  are input points,  $\mathcal{H}_{tl}$  denotes the  $l^{\text{th}}$  node in  $\mathcal{H}_t$ ,  $\mathcal{H}_{tl(a,b)}$  is the smallest (i.e. lowest) node in  $\mathcal{H}_t$  that contains both  $a$  and  $b$  and  $|\mathcal{H}_{tl(a,b)}|$  represents the number of training points (out of the whole training set of size  $N$ ) contained in that node’s subtree. Pairs that share a leaf node are given a distance of 0 because they are maximally similar under  $\mathcal{H}_t$ , and the minimum size of leaf nodes is defined by parameter, not by data. Each non-leaf node  $\mathcal{H}_{tl}$  is assigned (via max-margin clustering) a projection function  $\mathcal{P}_{tl}$  and associated binary linear discriminant  $\mathcal{S}_{tl}$  that divides the data in that node between the two child nodes.  $p_t(a, b)$  is a certainty term determined by the distance of the projected points  $a$  and  $b$  from the decision hyperplane at  $\mathcal{H}_{tl(a,b)}$ :

$$p_t(a, b) = \frac{1}{1 + \exp(\alpha \cdot \mathcal{P}_{tl(a,n)}(x_a))} - \frac{1}{1 + \exp(\alpha \cdot \mathcal{P}_{tl(a,b)}(x_b))} , \quad (3)$$

where  $\alpha$  is a hyperparameter that controls the sensitivity of  $p$ . Thus,  $p$  ranges from 0 to 1, approaching 0 when the projections of both  $a$  and  $b$  are near the decision boundary, and 1 when both are far away. The full distance formulation for a hierarchy is also confined to this range, with a distance approaching 1 corresponding to points that are widely separated at the root node, and 0 to points that share a leaf node.

## 2.2 HFD learning and inference

The fact that the trees used in HFD represent cluster hierarchies rather than decision trees has significant implications for HFD training, imposing stricter requirements on the learned splitting functions. While the goal of decision tree learning is ultimately to yield a set of pure single-class leaf nodes, a cluster hierarchy instead seeks to accurately group data elements at *every* level of the tree. Thus, if the hierarchy learning algorithm divides the data poorly at or near the root node, there is no way for it to recover from this error later on. This is partially mitigated by learning a forest in place of a single tree, but even in this case the *majority* of hierarchies in the forest must correctly model the high-level semantic relationship between any two data elements.

For this reason, HFD requires a robust approach to the hierarchy splitting problem that reliably generates semantically meaningful splits. Additionally, in order to allow for efficient metric inference, our splitting algorithm must generate explicit and efficiently evaluable splitting functions at each node.

Given these constraints, we approach the hierarchy learning problem as a series of increasingly fine-grained flat semi-supervised clustering problems, and we solve these flat clustering problems via max-margin clustering (MMC) [17, 18, 19, 20]. Max-margin clustering has a number of advantages that make it ideal for our problem:

- max-margin and large-margin methods have been proven effective in the metric learning domain [4, 21, 22]
- MMC returns a simple and explicit splitting function which can be applied to points outside the initial clustering
- MMC (including semi-supervised MMC) can be solved in linear time [20, 23, 24]

We employ a novel relaxed form of semi-supervised MMC, which uses pairwise must-link (ML) and cannot-link (CL) constraints to improve semantic clustering performance. Constraints of this type indicate either semantic similarity (ML) or dissimilarity (CL) between pairs of points, and can be provided by themselves.

We describe our semi-supervised MMC technique in Section 3.

### 2.2.1 Training algorithm

We train each tree in the HFD model independently, with each tree using the same data and constraint sets. Training is hence easily parallelized. Assume an unlabeled training

dataset  $\mathbf{X}^0$  and pairwise constraint set  $\mathcal{L}^0$ . Denote a must-link constraint set  $\mathcal{L}_{\mathcal{M}}^0$  and cannot-link constraint set  $\mathcal{L}_{\mathcal{C}}^0$ , such that  $\mathcal{L}^0 = \mathcal{L}_{\mathcal{M}}^0 \cup \mathcal{L}_{\mathcal{C}}^0$ .

Training of individual trees proceeds in a top-down manner. At each node  $\mathcal{H}_{tl}$  we begin by selecting a local feature subset  $\mathcal{K}_{tl}$  by uniformly sampling  $d_k < d$  features from the full feature set. We then replace each  $\mathbf{x}_j \in \mathbf{X}^{tl}$  with  $\begin{bmatrix} \mathbf{x}_j^{\mathcal{K}_{tl}} & 1 \end{bmatrix} \in \mathbf{X}_{\mathcal{K}}^{tl}$ .

For each node  $\mathcal{H}_{tl}$ , our split function learning algorithm can operate in either a semi-supervised or unsupervised mode, so before we begin learning we must check for constraint availability. We require at least 1 cannot-link constraint in order to carry out semi-supervised MMC, so we check whether  $\mathcal{L}_{\mathcal{C}}^{tl} = \emptyset$ , and then apply either semi-supervised or unsupervised MMC (see Section 3) to  $\mathbf{X}_{\mathcal{K}}^{tl}$  and  $\mathcal{L}^{tl}$ . The output of our split learning algorithm is the weight vector  $\mathbf{w}_{tl}$ , which, along with  $\mathcal{K}_{tl}$ , forms the splitting function  $\mathcal{S}_{tl}$ :

$$\mathcal{P}_{tl}(\mathbf{x}) = \mathbf{w}_{tl}^T \begin{bmatrix} \mathbf{x}_j^{\mathcal{K}_{tl}} & 1 \end{bmatrix} \quad (4)$$

$$\mathcal{S}_{tl}(\mathbf{x}) = \begin{cases} \text{send } \mathbf{x} \text{ left} & \mathcal{P}_{tl}(\mathbf{x}) \leq 0 \\ \text{send } \mathbf{x} \text{ right} & \mathcal{P}_{tl}(\mathbf{x}) > 0 \end{cases} \quad (5)$$

We then apply  $\mathcal{S}_{tl}$  to divide  $\mathbf{X}_{tl}$  among  $\mathcal{H}_{tl}$ 's children. After this, we must also propagate the constraints down the tree. We do this by iterating through  $\mathcal{L}_{tl}$  and checking the point membership of each child node  $\mathcal{H}_{tj}$ —if  $\mathbf{X}_{tj}$  contains both points covered by a constraint, then we add that constraint to  $\mathcal{L}_{tj}$ .

As a result, constraints in  $\mathcal{L}_{tl}$  whose constrained points are separated by  $\mathcal{H}_{tl}$ 's splitting function effectively disappear in the next level of the hierarchy. This results in a steady narrowing of the constraint-satisfaction problem as we reach further down the tree, in accordance with the progressively smaller regions of the data space we are processing. We continue this process until we reach a stopping point (in our experiments, a minimum node size threshold), falling back on unsupervised MMC as we exhaust the relevant cannot-link constraints.

### 2.2.2 Inference

Metric inference on learned HFD structures is straightforward. We feed two points  $\mathbf{x}_1$  and  $\mathbf{x}_2$  to the metric and track their progress down each tree  $\mathcal{H}_t$ . At each node  $\mathcal{H}_{tl}$ , we compute  $\mathcal{S}_{tl}(\mathbf{x}_1)$  and  $\mathcal{S}_{tl}(\mathbf{x}_2)$ . If  $\mathcal{S}_{tl}(\mathbf{x}_1) = \mathcal{S}_{tl}(\mathbf{x}_2)$ , we continue the process in the indicated child node. If not, then we have found  $\mathcal{H}_{tl}(\mathbf{x}_1, \mathbf{x}_2)$ , so we compute and return  $\mathcal{H}_t(\mathbf{x}_1, \mathbf{x}_2)$  as described in (2). The results from each tree are then combined as per (1).

## 3 Learning splitting functions

In order to learn strong, optimized splitting functions at each hierarchy node, our method relies on the Max-Margin Clustering (MMC) framework. In most nodes, our method uses semi-supervised MMC (SSMMC) to incorporate pairwise constraint information into the split function learning

process. Below, we describe a state-of-the art SSMMC formulation, as well as our own novel modifications to this formulation that allow it to function in our hierarchical problem setting.

### 3.1 Semi-supervised max-margin clustering

SSMMC incorporates a set of must-link ( $\mathcal{L}_M$ ) and cannot-link ( $\mathcal{L}_C$ ) pairwise semantic constraints into the clustering problem. Thus, where unsupervised MMC seeks only to maximize the cluster assignment margin of each point, SSMMC includes an additional set of margin terms reflecting the satisfaction of each pairwise constraint.

For convenience, first define the following function representing the joint projection value of two different points onto two particular cluster labels<sup>1</sup>:

$$\phi(\mathbf{x}_1, \mathbf{x}_2, y_1, y_2) = y_1 \mathbf{w}^T \mathbf{x}_1 + y_2 \mathbf{w}^T \mathbf{x}_2 \quad (6)$$

The semi-supervised MMC problem [24] is then formulated as:

$$\begin{aligned} \min_{\mathbf{w}, \eta, \xi} \quad & \frac{\lambda}{2} \|\mathbf{w}\|^2 + \frac{1}{L_M + L_C} \left( \sum_{j \in \mathcal{L}_M} \eta_j + \sum_{j \in \mathcal{L}_C} \eta_j \right) + \frac{C}{U} \sum_{i=1}^U \xi_i \\ \text{s.t.} \quad & \forall j \in \mathcal{L}_M, \forall s_{j1}, s_{j2} \in \{-1, 1\}, s_{j1} \neq s_{j2} : \\ & \quad \max_{z_{j1}=z_{j2}} \phi(\mathbf{x}_{j1}, \mathbf{x}_{j2}, z_{j1}, z_{j2}) - \phi(\mathbf{x}_{j1}, \mathbf{x}_{j2}, s_{j1}, s_{j2}) \\ & \quad \geq 1 - \eta_j, \quad \eta_j \geq 0 \\ & \forall j \in \mathcal{L}_C, \forall s_{j1}, s_{j2} \in \{-1, 1\}, s_{j1} = s_{j2} : \\ & \quad \max_{z_{j1} \neq z_{j2}} \phi(\mathbf{x}_{j1}, \mathbf{x}_{j2}, z_{j1}, z_{j2}) - \phi(\mathbf{x}_{j1}, \mathbf{x}_{j2}, s_{j1}, s_{j2}) \\ & \quad \geq 1 - \eta_j, \quad \eta_j \geq 0 \\ & \forall i \in \mathcal{U} : \\ & \quad \max_{y_i^s \in \{-1, 1\}} 2y_i^s \mathbf{w}^T \mathbf{x}_i \geq 1 - \xi_i, \\ & \quad \xi_i \geq 0, \end{aligned} \quad (7)$$

where  $\mathcal{L}_M$  and  $\mathcal{L}_C$  are the sets of ML and CL constraints, respectively,  $L_M$  and  $L_C$  are the sizes of those sets,  $\eta_j$  are slack variables for the pairwise constraints,  $\mathcal{U}$  is the set of unconstrained points,  $U$  is the size of that set and  $\xi_i$  are slack variables for the unconstrained points.  $j_1$  and  $j_2$  represent the two points constrained by pairwise constraint  $j$ . Here, the must-link and cannot-link constraints each impose a soft margin on the difference in score between the highest-scoring joint projection that satisfies the constraint and the highest scoring joint projection that does *not* satisfy the constraint. This formulation is sufficient for standard clustering problems, but requires some modification in order to function well in our problem setting.

<sup>1</sup>Note that we will periodically refer to  $\phi_t$  or  $\phi_r$ —these simply indicate that the  $\phi$  function is using the temporary  $\mathbf{w}$  value at the given iteration (e.g.  $\mathbf{w}^{(t)}$ ).

### 3.2 Relaxed semi-supervised max-margin clustering

Because cannot-link constraints disappear from the hierarchy learning problem once they are satisfied (see Section 2.2.1), the number of relevant cannot-link constraints will generally decrease much more quickly than the number of must-link constraints. This will lead to highly imbalanced constraint sets in lower levels of the hierarchy.

Under the original SSMMC formulation, imbalanced cases such as these may well yield trivial one-class solutions wherein the ML constraints are well satisfied, but the few CL constraints are highly *unsatisfied*. To address this problem, we simply separate the ML and CL constraints into two distinct optimization terms, each with equal weight.

Second, and more significantly, we must modify SSMMC to handle the *hierarchical* nature of our problem. Consider a case with 4 semantic classes: apple, orange, bicycle and motorcycle. In a binary hierarchical setting, the most reasonable way to approach this problem is to first separate apples and oranges from bicycles and motorcycles, then divide the data into pure leaf nodes lower in the tree.

Standard SSMMC, however, will instead attempt to simultaneously satisfy the cannot-link constraints between *all* of these classes, which is impossible. As a result, the optimization algorithm may seek a compromise solution that weakly violates all or most of the constraints, rather than one that strongly satisfies a subset of the constraints and ignores the others (e.g. that separates apples and oranges from bicycles and motorcycles).

We handle this complication by relaxing the clustering algorithm to focus on only a *subset* of the CL constraint set, and integrate the selection of that subset into the optimization problem. Thus, our variant of semi-supervised MMC simultaneously optimizes  $\mathbf{w}$  to satisfy a subset of the CL constraint set  $\mathcal{L}'_C \subset \mathcal{L}_C$ , and seeks the  $\mathcal{L}'_C$  that can best be satisfied by a binary linear discriminant:

$$\begin{aligned} \min_{\mathbf{w}, \eta, \xi, \mathcal{L}'_C} \quad & \frac{\lambda}{2} \|\mathbf{w}\|^2 + \frac{1}{L_M} \sum_{j \in \mathcal{L}_M} \eta_j + \frac{1}{L'_C} \sum_{j \in \mathcal{L}'_C} \eta_j + \frac{C}{U} \sum_{i=1}^U \xi_i \\ \text{s.t.} \quad & \forall j \in \mathcal{L}_M, \forall s_{j1}, s_{j2} \in \{-1, 1\}, s_{j1} \neq s_{j2} : \\ & \quad \max_{z_{j1}=z_{j2}} \phi(\mathbf{x}_{j1}, \mathbf{x}_{j2}, z_{j1}, z_{j2}) - \phi(\mathbf{x}_{j1}, \mathbf{x}_{j2}, s_{j1}, s_{j2}) \\ & \quad \geq 1 - \eta_j, \quad \eta_j \geq 0 \\ & \exists \mathcal{L}'_C \subset \mathcal{L}_C \text{ of size } L'_C \text{ s.t. :} \\ & \quad \forall j \in \mathcal{L}'_C, \forall s_{j1}, s_{j2} \in \{-1, 1\}, s_{j1} = s_{j2} : \\ & \quad \quad \max_{z_{j1} \neq z_{j2}} \phi(\mathbf{x}_{j1}, \mathbf{x}_{j2}, z_{j1}, z_{j2}) - \phi(\mathbf{x}_{j1}, \mathbf{x}_{j2}, s_{j1}, s_{j2}) \\ & \quad \quad \geq 1 - \eta_j, \quad \eta_j \geq 0 \\ & \forall i \in \mathcal{U} : \\ & \quad \max_{y_i^s \in \{-1, 1\}} 2y_i^s \mathbf{w}^T \mathbf{x}_i \geq 1 - \xi_i, \\ & \quad \xi_i \geq 0, \end{aligned} \quad (8)$$

We set the size of  $\mathcal{L}'_C$  via the parameter  $L'_C$ .

### 3.2.1 Semi-supervised MMC optimization

We optimize our SSMMC formulation via a Constrained Concave Convex Procedure (CCCP) [25, 24]. This is an iterative process that, at each iteration  $t$ , frames the constraints in (8) as the difference of two convex functions and replaces one of those functions with its tangent at  $\mathbf{w}^{(t)}$ , resulting in a convex optimization problem than can be easily solved via subgradient projection.

We first use a heuristic [24] to initialize  $\mathbf{w}^{(0)}$  non-randomly based on  $\mathcal{L}$ . We compute pairwise constraint scatter matrices  $\mathbf{S}_{\mathcal{M}}$  and  $\mathbf{S}_{\mathcal{C}}$ :

$$\mathbf{S}_{\mathcal{M}} = \frac{1}{L_{\mathcal{M}}} \sum_{j \in \mathcal{L}_{\mathcal{M}}} (\mathbf{x}_{j1} - \mathbf{x}_{j2})(\mathbf{x}_{j1} - \mathbf{x}_{j2})^T \quad (9)$$

$$\mathbf{S}_{\mathcal{C}} = \frac{1}{L_{\mathcal{C}}} \sum_{j \in \mathcal{L}_{\mathcal{C}}} (\mathbf{x}_{j1} - \mathbf{x}_{j2})(\mathbf{x}_{j1} - \mathbf{x}_{j2})^T. \quad (10)$$

We can then use  $\mathbf{S}_{\mathcal{M}}$  and  $\mathbf{S}_{\mathcal{C}}$  to compute a projection  $\mathbf{w}^{(0)}$  that attempts to maximize distance between CL and minimize distance between ML point pairs by computing the largest eigenvector of the general eigenproblem:

$$\mathbf{S}_{\mathcal{C}} \mathbf{v} = \lambda \mathbf{S}_{\mathcal{M}} \mathbf{v}. \quad (11)$$

Note that, because  $\mathcal{L}'_{\mathcal{C}}$  is not available at this point in the optimization, this step must be performed on all of the cannot-link constraints.

Once  $\mathbf{w}^{(0)}$  has been computed, we can begin CCCP to optimize the max-margin problem. Given  $\mathbf{w}^{(t)}$ , we begin each iteration of CCCP by computing the tangents of constraint functions at  $\mathbf{w}^{(t)}$ :

Denote  $(z_{j1}^{\mathcal{M}(t)}, z_{j2}^{\mathcal{M}(t)})$  and  $(z_{j1}^{\mathcal{C}(t)}, z_{j2}^{\mathcal{C}(t)})$  as the best cluster assignments under  $\mathbf{w}^{(t)}$  that satisfy their associated constraint  $j$ :

$$(z_{j1}^{\mathcal{M}(t)}, z_{j2}^{\mathcal{M}(t)}) = \underset{z_{j1}=z_{j2}|j \in \mathcal{L}_{\mathcal{M}}}{\operatorname{argmax}} \phi_t(\mathbf{x}_{j1}, \mathbf{x}_{j2}, z_{j1}, z_{j2}) \quad (12)$$

$$(z_{j1}^{\mathcal{C}'(t)}, z_{j2}^{\mathcal{C}'(t)}) = \underset{z_{j1} \neq z_{j2}|j \in \mathcal{L}'_{\mathcal{C}}}{\operatorname{argmax}} \phi_t(\mathbf{x}_{j1}, \mathbf{x}_{j2}, z_{j1}, z_{j2}) \quad (13)$$

Similarly, denote  $y_i^{(t)}$  as the best unary cluster assignment label for  $\mathbf{x}_i$  under  $\mathbf{w}^{(t)}$ :

$$y_i^{(t)} = \underset{y_i \in \{-1, 1\}}{\operatorname{argmax}} y_i \mathbf{w}^{(t)T} \mathbf{x}_i \quad (14)$$

Finally, we select a candidate  $\mathcal{L}'_{\mathcal{C}}^{(t)}$  by choosing the  $L'_{\mathcal{C}}$  cannot-link constraints with the largest satisfaction margin (again, under  $\mathbf{w}^{(t)}$ ):

$$\mathcal{L}'_{\mathcal{C}}^{(t)} = \underset{\mathcal{L}'_{\mathcal{C}} \subset \mathcal{L}_{\mathcal{C}}, |\mathcal{L}'_{\mathcal{C}}| = L'_{\mathcal{C}}}{\operatorname{argmax}} \sum_{j \in \mathcal{L}'_{\mathcal{C}}} \min_{s_{j1}=s_{j2}} \left[ \phi_t(\mathbf{x}_{j1}, \mathbf{x}_{j2}, z_{j1}^{\mathcal{C}(t)}, z_{j2}^{\mathcal{C}(t)}) - \phi_t(\mathbf{x}_{j1}, \mathbf{x}_{j2}, s_{j1}, s_{j2}) \right] \quad (15)$$

By setting all of these values as constants, we obtain a

convex (but non-differentiable) optimization problem:

$$\begin{aligned} \min_{\mathbf{w}^{(t)}, \eta, \xi} \quad & \frac{\lambda}{2} \|\mathbf{w}^{(t)}\|^2 + \frac{1}{L_{\mathcal{M}}} \sum_{j \in \mathcal{L}_{\mathcal{M}}} \eta_j + \frac{1}{L'_{\mathcal{C}}} \sum_{j \in \mathcal{L}'_{\mathcal{C}}^{(t)}} \eta_j + \frac{C}{U} \sum_{i=1}^U \xi_i \\ \text{s.t.} \quad & \forall j \in \mathcal{L}_{\mathcal{M}}, \forall s_{j1}, s_{j2} \in \{-1, 1\}, s_{j1} \neq s_{j2} : \\ & \phi_t(\mathbf{x}_{j1}, \mathbf{x}_{j2}, z_{j1}^{\mathcal{M}(t)}, z_{j2}^{\mathcal{M}(t)}) - \phi_t(\mathbf{x}_{j1}, \mathbf{x}_{j2}, s_{j1}, s_{j2}) \\ & \geq 1 - \eta_j, \quad \eta_j \geq 0 \\ & \forall j \in \mathcal{L}'_{\mathcal{C}}^{(t)}, \forall s_{j1}, s_{j2} \in \{-1, 1\}, s_{j1} = s_{j2} : \\ & \phi_t(\mathbf{x}_{j1}, \mathbf{x}_{j2}, z_{j1}^{\mathcal{C}'(t)}, z_{j2}^{\mathcal{C}'(t)}) - \phi_t(\mathbf{x}_{j1}, \mathbf{x}_{j2}, s_{j1}, s_{j2}) \\ & \geq 1 - \eta_j, \quad \eta_j \geq 0 \\ & \forall i \in \mathcal{U} : \\ & 2y_i^{(t)} \mathbf{w}^{(t)T} \mathbf{x}_i \geq 1 - \xi_i, \\ & \xi_i \geq 0, \end{aligned} \quad (16)$$

This problem can be efficiently solved via subgradient projection. At each subgradient iteration  $r$ ,  $\nabla_r$  can be computed via:

$$\begin{aligned} \nabla_r = \quad & \lambda \mathbf{w} \\ & + \frac{1}{L_{\mathcal{M}}} \sum_{j \in \mathcal{L}_{\mathcal{M}}^{(r)}} \left[ \left( s_{j1}^{\mathcal{M}^r} x_{j1} + s_{j2}^{\mathcal{M}^r} x_{j2} \right) \right. \\ & \left. - \left( z_{j1}^{\mathcal{M}(t)} x_{j1} + z_{j2}^{\mathcal{M}(t)} x_{j2} \right) \right] \\ & + \frac{1}{L'_{\mathcal{C}}} \sum_{j \in \mathcal{L}'_{\mathcal{C}}^{(r)}} \left[ \left( s_{j1}^{\mathcal{C}'^r} x_{j1} + s_{j2}^{\mathcal{C}'^r} x_{j2} \right) \right. \\ & \left. - \left( z_{j1}^{\mathcal{C}'(t)} x_{j1} + z_{j2}^{\mathcal{C}'(t)} x_{j2} \right) \right] \\ & + \frac{C}{U} \sum_{i \in \mathcal{U}} y_i^{(t)} \mathbf{x}_i \mathbb{1} \left( 2y_i^{(t)} \mathbf{w}^T \mathbf{x}_i \leq 1 \right), \end{aligned} \quad (17)$$

where  $\mathcal{L}_{\mathcal{M}}^{(r)}$  and  $\mathcal{L}'_{\mathcal{C}}^{(r)}$  are the set of pairwise constraints with non-zero  $\eta_j$  values under  $\mathbf{w}^{(r)}$ :

$$\mathcal{L}_{\mathcal{M}}^{(r)} = \left\{ j \in \mathcal{L}_{\mathcal{M}} \mid \phi_r(\mathbf{x}_{j1}, \mathbf{x}_{j2}, z_{j1}^{\mathcal{M}(t)}, z_{j2}^{\mathcal{M}(t)}) - \max_{s_{j1} \neq s_{j2}} \phi_r(\mathbf{x}_{j1}, \mathbf{x}_{j2}, s_{j1}, s_{j2}) < 1 \right\} \quad (18)$$

$$\mathcal{L}'_{\mathcal{C}}^{(r)} = \left\{ j \in \mathcal{L}'_{\mathcal{C}} \mid \phi_r(\mathbf{x}_{j1}, \mathbf{x}_{j2}, z_{j1}^{\mathcal{C}'(t)}, z_{j2}^{\mathcal{C}'(t)}) - \max_{s_{j1} = s_{j2}} \phi_r(\mathbf{x}_{j1}, \mathbf{x}_{j2}, s_{j1}, s_{j2}) < 1 \right\}, \quad (19)$$

and we define the highest-scoring constraint-violating assignments for each pair:

$$(s_{j1}^{\mathcal{M}^r}, s_{j2}^{\mathcal{M}^r}) = \underset{j \in \mathcal{L}_{\mathcal{M}}, s_{j1} \neq s_{j2}}{\operatorname{argmax}} \phi_r(\mathbf{x}_{j1}, \mathbf{x}_{j2}, s_{j1}, s_{j2}) \quad (20)$$

$$(s_{j1}^{\mathcal{C}'^r}, s_{j2}^{\mathcal{C}'^r}) = \underset{j \in \mathcal{L}'_{\mathcal{C}}, s_{j1} = s_{j2}}{\operatorname{argmax}} \phi_r(\mathbf{x}_{j1}, \mathbf{x}_{j2}, s_{j1}, s_{j2}). \quad (21)$$

Finally, it is shown in [24] that the optimal solution to  $\mathbf{w}$  is bounded by:

$$\mathbf{w}^* \in \left\{ \mathbf{w} \mid \|\mathbf{w}\| \leq \rho = \sqrt{\frac{1+C}{\lambda}} \right\}, \quad (22)$$

**Algorithm 1** Subgradient optimization of  $\mathbf{w}$  in semi-supervised MMC

---

```

 $r \leftarrow 0$ 
randomly initialize  $\mathbf{w}^{(r)}$ , s.t.  $\|\mathbf{w}^{(r)}\| \leq \rho$ 
while  $\mathbf{w}^{(r)}$  not converged do
  Compute  $\nabla_r$  via (17)
   $\mathbf{w}^{(r+\frac{1}{2})} \leftarrow \mathbf{w}^{(r)} + \frac{1}{\lambda r} \nabla_r$ 
   $\mathbf{w}^{(r+1)} \leftarrow \min \left( 1, \rho / \|\mathbf{w}^{(r+\frac{1}{2})}\| \right) \mathbf{w}^{(r+\frac{1}{2})}$ 
  if  $\|\mathbf{w}^{(r+1)} - \mathbf{w}^{(r)}\| / \max(\|\mathbf{w}^{(r)}\|, \|\mathbf{w}^{(r+1)}\|) \leq \epsilon_1$  then
     $\mathbf{w}^{(r)}$  is converged
  end if
   $r \leftarrow r + 1$ 
end while

```

---

so we project  $\mathbf{w}^{(r)}$  back into this space at each iteration.

The subgradient optimization method for solving (16) is described in Algorithm 1 and the full CCCP procedure for semi-supervised MMC is described in Algorithm 2 (note that we set  $C = 0$  for the first three iterations in order to allow the more reliable supervised constraints to guide the optimization to a strong solution region, before introducing the unsupervised constraints to refine it).

**Algorithm 2** CCCP optimization for semi-supervised MMC

---

```

 $t \leftarrow 0$ 
randomly initialize  $\mathbf{w}^{(t)}$ , s.t.  $\|\mathbf{w}^{(t)}\| \leq \rho$ 
while  $\mathbf{w}^{(t)}$  not converged do
  if  $t \nmid 3$  then
     $C^{(t)} \leftarrow 0$ 
  else
     $C^{(t)} \leftarrow C$ 
  end if
  Compute  $y^{(t)}$  via (14)
  Compute  $(z_{j1}^{\mathcal{M}(t)}, z_{j2}^{\mathcal{M}(t)})$  via (12)
  Compute  $(z_{j1}^{C(t)}, z_{j2}^{C(t)})$  via (13)
  Compute  $\mathcal{L}_C'^{(t)}$  via (15)
  Compute  $w^{(t+1)}$  via Algorithm 1
  if  $\|\mathbf{w}^{(t+1)} - \mathbf{w}^{(t)}\| / \max(\|\mathbf{w}^{(t)}\|, \|\mathbf{w}^{(t+1)}\|) \leq \epsilon_2$  then
     $\mathbf{w}^{(t)}$  is converged
  end if
   $t \leftarrow t + 1$ 
end while

```

---

### 3.3 Unsupervised MMC

Because HFD progressively eliminates constraints as the hierarchy grows deeper (see Section 2.2.1), at lower levels of the hierarchy we may encounter nodes where there are no cannot-link constraints available, and hence SSMMC cannot proceed [24]. In these cases, we fall back on the unsupervised Membership Requirement MMC (MRMMC) formulation proposed by Hoai and De la Torre [20]. We optimize this problem via block-coordinate descent as described in that work.

## 4 Fast approximate nearest neighbors in hierarchy metric space

One problem with this approach is the potentially high (though still embarrassingly parallel) cost of computing each pairwise distance, as compared to a Euclidean or even Mahalanobis distance. This is worsened, for many applications, by the unavailability of traditional fast approximate nearest-neighbor methods (e.g. kd-trees [26], hierarchical  $k$ -means [27] or hashing [28]), which require an explicit representation of the data in the metric space in order to function.

We address the latter problem by introducing our own fast approximate nearest-neighbor algorithm, which takes advantage of the tree-based structure of the metric to greatly reduce the number of pairwise distance computations needed to compute a set of nearest-neighbors for a query point  $x$ .

We begin by tracing the path taken by  $x$  through each tree in the forest, and thus identifying each leaf node containing  $x$ . We then seek  $k_{\mathcal{O}}$  candidate neighbors from each tree, beginning by sampling other training points from the identified leaf nodes, then, if necessary, moving up the tree parent-node-by-parent-node until  $k_{\mathcal{O}}$  candidates have been found. The candidate sets from each tree are then combined to yield a final candidate neighbor set  $\mathcal{O}$ , such that  $|\mathcal{O}| \leq T \cdot k_{\mathcal{O}}$ . We then compute the full hierarchy distance  $D(x, y)$  for all  $y \in \mathcal{O}$ , sort the resulting distances, and return the  $k$  closest points.

This approximation method functions by assuming that, intuitively, a point's nearest-neighbors within the full forest metric space are very likely to *also* be nearest-neighbors under at least *one* of the individual tree metrics. We evaluate this method empirically on several small-to-midsized datasets, and the results strongly support the validity of this approximation (Section 6.2).

## 5 Complexity Analysis

### HFD training

The overall complexity of semi-supervised max-margin clustering is  $\mathbf{O}(d^3 + nd)$  [24] (where  $n$  is the total number of constraints plus the number of unconstrained points). If we ignore  $d$  (which in our case is replaced by the parameter  $d_k$ , and generally speaking  $d_k \ll d$ ), this leaves SSMMC as a  $\mathbf{O}(n)$  operation. In the tree setting, we are using SSMMC in a divide-and-conquer fashion—thus, if we assume that we divide the data roughly in half with each SSMMC operation, the complexity of fully training an HFD tree is  $\mathbf{O}(n \log n)$ , and the total training cost is thus  $\mathbf{O}(Tn \log n)$ . Given the embarrassingly parallel nature of the problem, the  $T$  factor can be ignored in many cases, allowing an HFD model to be trained in  $\mathbf{O}(n \log n)$  time.

### HFD inference

Computing a single HFD metric distance requires traversing down each tree in the forest one time, for an (again embarrassingly parallel) complexity cost of  $\mathbf{O}(T \log n)$ . Many of the most common applications of a metric require computing nearest-neighbors between the training set and a

test set of size  $m$ . This requires  $mn$  distance evaluations, so a brute force nearest-neighbor search under HFD costs  $\mathbf{O}(mnT \log n)$ , or  $\mathbf{O}(nT \log n)$  for a single test point.

Our approximate nearest-neighbor algorithm significantly reduces this cost. Computing candidate neighbors for a single point costs only  $\mathbf{O}(T \log n)$ . There will be at most  $Tk_{\mathcal{O}}$  candidates for each set, so the cost for computing distances to each candidate is  $\mathbf{O}(T^2 k_{\mathcal{O}} \log n)$ , or  $\mathbf{O}(T^2 \log n)$  if we ignore the parameter. It should be noted, though, that in practice there is significant overlap between the candidate sets returned by different trees, and this overlap increases with  $T$ , so the actual cost of this step is generally much lower.

Thus, the complexity of our approximate nearest-neighbor method, when applied to an entire dataset, is  $\mathbf{O}(mT^2 \log n)$ . Even in the worst case, this is an improvement (since  $T \ll n$  on even moderately sized datasets), and in practice is generally much better than the worst case.

## 6 Experiments

Below we present several experiments quantifying HFD's performance. First, we validate the accuracy and efficiency of our approximate nearest-neighbor retrieval method. We then carry out benchmark comparisons against other state-of-the-art metric learning techniques in the  $k$ -nearest neighbor classification, large-scale image retrieval and semi-supervised clustering domains.

### 6.1 Datasets

We use a range of datasets, from small- to large-scale, to evaluate our method. For small to mid-range data, we use a number of well-known UCI sets [29]: sonar (208 x 60 x 2), ionosphere (351 x 34 x 2), balance (625 x 4 x 3), segmentation (2,310 x 18 x 7) and magic (19,020 x 10 x 2), as well as the USPS handwritten digits dataset (11,000 x 256 x 10) [30].

For our larger scale experiments, we relied on the CIFAR tiny image datasets [31]. CIFAR-10 consists of 50,000 training and 10,000 test images, spread among 10 classes. CIFAR-100 also contains 50,000 training and 10,000 testing images, but has 2 different label sets—a coarse set with 20 classes, and a fine set with 100 classes. All CIFAR instances are 36x36 color images, which we have reduced to 300 features via PCA.

In all our experiments, the data is normalized to 0 mean and unit variance before any metric methods are applied to it.

### 6.2 Approximate nearest-neighbor retrieval

Because we use it for retrieval in all of our other experiments, we first evaluate the accuracy cost and efficiency benefits of our approximate nearest-neighbor method. We evaluate accuracy by training an HFM model with 100 trees. We then return 50 approximate nearest-neighbors for each point in the dataset and compute mean average precision

(mAP) relative to the ground truth 50 nearest-neighbors (obtained via brute force search). Average precision scores are computed at 10, 20, 30, 40 and 50 nearest-neighbors. We do retrieval at  $k_{\mathcal{O}} = 1, 3, 5, 10, 20$  and 30, and report both the mAP results and the time taken (as a proportion of the brute force time) at each value on several datasets.

Table 1: Approximate nearest-neighbor retrieval mAP scores

$k_{\mathcal{O}}$	Sonar	Seg.	USPS
1	0.812	0.715	0.6559
3	0.965	0.904	0.8700
5	0.987	0.956	0.9274
10	0.997	0.987	0.9710
20	0.998	0.994	0.9897
30	0.998	0.995	0.9945

Table 2: Approximate nearest-neighbor retrieval times (as a proportion of brute force search time)

$k_{\mathcal{O}}$	Sonar	Seg.	USPS
1	0.499	0.042	0.014
3	0.547	0.074	0.034
5	0.729	0.097	0.049
10	0.860	0.147	0.076
20	1.002	0.221	0.112
30	0.997	0.270	0.136

The results clearly show that our approximation method can yield significant reductions in retrieval time on larger datasets, and does so with minimal loss of accuracy. Note that all other results we report for HFD are generated using this approximation method. We use  $k_{\mathcal{O}} = 5$  for the CIFAR datasets, and  $k_{\mathcal{O}} = 10$  for all other data.

### 6.3 Comparison methods and parameters

In the following experiments, we compare our HFD model against a number of state-of-the-art metric learning techniques: DCA [32], LMNN [4, 22], GB-LMNN [12], ITML [2], Boostmetric [3] and RFD [13]. With the exception of RFD and GB-LMNN (both of which incorporate tree structures into their metrics), all are Mahalanobis methods that learn purely linear transforms of the original data.

For all ITML experiments, we cross-validated across 5 different  $\gamma$  values and reported the best results.

We did not extensively tune any of the hyperparameters for HFD, instead using a common set of values (or rules for assigning values) for all datasets. We set  $T = 500$  (for HFD, RFD and GB-LMNN),  $d_k = \frac{d}{3}$  (with the exception of the balance dataset, where we use  $d_k = d$ ),  $L'_C = 0.25L_C$ ,  $\lambda = 0.01$ ,  $C = 1$ ,  $\epsilon_1 = \epsilon_2 = 0.01$  and  $\alpha = 0.5$ . As a stop criteria for tree training, we set a minimum node size of 5 for the clustering and classification experiments, and 30 for the retrieval experiments.

### 6.4 Nearest neighbor classification

We next test our method using  $k$ -nearest neighbor classification (we use  $k = 5$  for all datasets). Each dataset

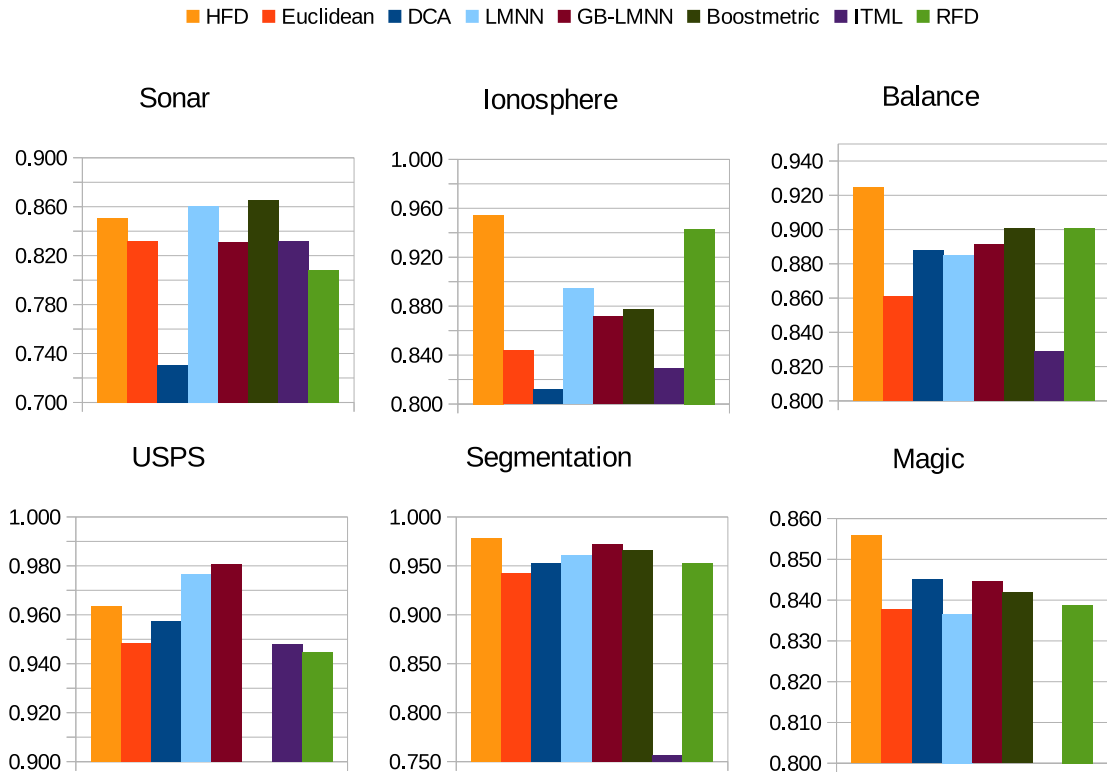


Figure 1: 5-nearest neighbor classification accuracy under HFD and benchmark metrics. HFD achieves the highest accuracy among tested methods on 4 out of 6 datasets, and is competitive on the remaining two. (View in color)

is evaluated using 5-fold cross-validation. For the weakly-supervised methods, in each fold we use 1,000 constraints per class (drawn from the training data) for the sonar, ionosphere, balance and segmentation datasets. For USPS and magic, we use 30,000 constraints in each fold. Our results are shown in Figure 1.

We found that HFD achieved the best score on 4 out of the 6 datasets tested, and was competitive on the remaining two (sonar and USPS).

We also performed some additional experiments to test the robustness of our approach compared to other methods. We tested 5-nearest neighbor classification on the sonar dataset with varying amounts of noise added to the training labels (for consistency, we used the same noise data for all metrics). Our results are shown in Figure 2.

While our method is only minimally effected by the added noise, the performance of all other metrics drops dramatically. Though HFD does not obtain the best results in the noiseless case, with just 10% of the training labels corrupted our results become significantly better than all other metrics. This effect is even more pronounced at 20% noise.

## 6.5 Retrieval

To evaluate our method’s performance (as well as, implicitly, the effectiveness of our approximate nearest-neighbor algorithm) on large-scale tasks, we computed semantic retrieval precision on labeled CIFAR tiny image datasets. For the weakly-supervised methods, we sample 600,000 constraints from the training data (which is less than 0.1% of

the full constraint set). We do not report Boostmetric results on these sets because we were unable to obtain them.

Our results can be found in Figure 3, which shows retrieval accuracy at 5 through 50 images retrieved on each dataset. HFD is clearly the best-performing method across all 3 problems. While DCA is competitive with HFD on the 10-class and 20-class sets, this performance drops off significantly on the more difficult 100-class problem.

The particularly strong performance of HFD on the 100-class problem may be due to the relaxed SSMMC formulation, which allows our method to effectively divide the very difficult 100-class discrimination problem into a sequence of many broader, easier problems, and thus make more effective use of its cannot-link constraints than the other metrics.

## 6.6 Semi-supervised clustering

In order to analyze the metrics holistically, in a way that takes into account not just ordered rankings of distances but the relative values of the distances themselves, we began by performing semi-supervised clustering experiments. We sampled varying numbers of constraints from each of the datasets presented and used these constraints to train the metrics. Note that only weakly- or semi-supervised metrics could be evaluated in this way, so only DCA, ITML, RFD and HFD were used in this experiment.

After training, the learned metrics were applied to the dataset and used to retrieve the 50 nearest-neighbors and corresponding distances for each point. RFD and HFD return distances on a 0-1 scale, so we converted those to



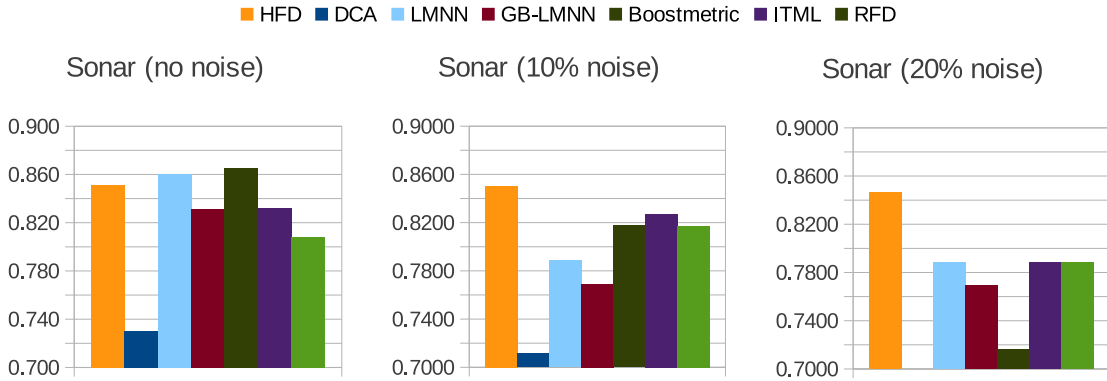


Figure 2: 5-nearest neighbor classification accuracy on the sonar dataset with varying levels of training label noise. While HFD does not achieve the best result in the noiseless case, it is far more robust to label noise than any other method tested. (View in color)

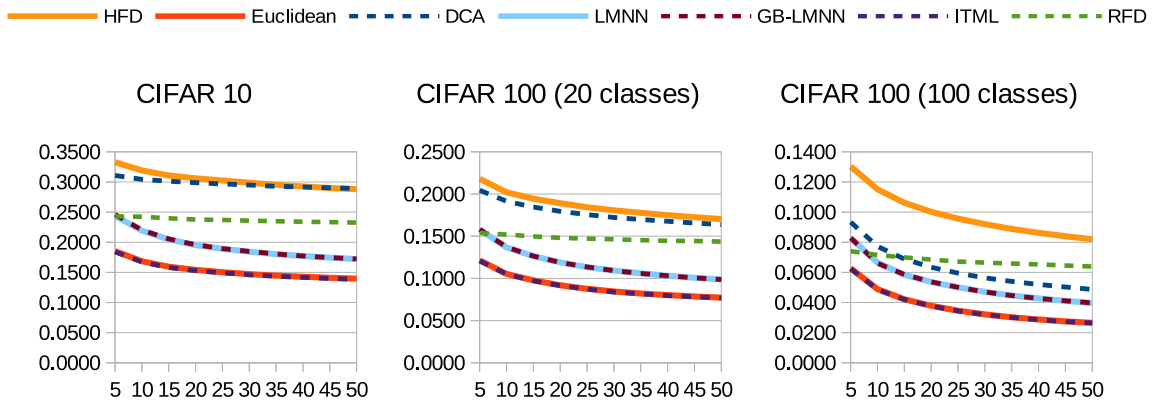


Figure 3: Large-scale semantic image retrieval results for our method and benchmarks. Only DCA is competitive with our method on the 10 and 20 class datasets, and HFD significantly outperforms all other algorithms on the 100 class problem. (View in color)

similarities by simply subtracting from 1. For the other methods, the distances were converted to similarities by applying a Gaussian kernel (we used  $\sigma = 0.1, 1, 10, 100$  or 1000—whichever yielded the best results for that metric and dataset).

We then used the neighbor and similarity data to construct a number of sparse similarity matrices from varying numbers of nearest-neighbors (ranging from 5 to 50) and computed a spectral clustering [33] solution for each. We evaluated these clustering outputs using V-Measure [34] and recorded the best result for each metric-dataset pair (see Table 3—the numbers below the dataset names indicate the number of constraints used in that test).

The tree based methods, RFD and HFD, demonstrated a consistent and significant advantage on this data. Between the two tree-based methods, HFD yielded better results on the sonar and balance data, while both were competitive on the segmentation and USPS datasets.

It is notable that the difference between the euclidean performance and that of the tree-based metrics is much more pronounced in the clustering domain. This would suggest that the actual distance values (as opposed to the distance

rankings) returned by the tree-based metrics contain much stronger semantic information than those returned by the linear methods.

## 7 Conclusion

In this paper, we have presented a novel semi-supervised nonlinear distance metric learning procedure based on forests of cluster hierarchies constructed via an iterative max-margin clustering procedure with a novel relaxed constraint formulation. Our experimental results show that that this algorithm is competitive with the state-of-the-art on small- and medium-scale datasets, and potentially superior for large-scale problems. We also present a novel in-metric approximate nearest-neighbor retrieval algorithm for our method that greatly decreases retrieval times for large data with little reduction in accuracy.

In the future, we hope to expand this metric to less-well-explored learning settings, such as those with more complex semantic relationship structures (e.g., hierarchies or “soft” class membership). By extending our method to incorporate relative similarity triplet constraints, we could al-

Table 3: Semi-supervised clustering results (V-Measure)

	Sonar			Balance		
	60	120	180	45	90	180
Euclidean	0.0493	0.0493	0.0493	0.2193	0.2193	0.2193
DCA	0.0959	0.1098	0.1386	0.0490	0.2430	0.3817
ITML	0.0650	0.0555	0.0644	0.2221	0.1915	0.2155
RFD	0.0932	0.1724	0.2699	0.1398	0.1980	0.3004
HFD	<b>0.1267</b>	<b>0.2296</b>	<b>0.3518</b>	<b>0.3059</b>	<b>0.5128</b>	<b>0.6149</b>

	Segmentation			USPS		
	70	175	350	3k	5k	10k
Euclidean	0.6393	0.6393	0.6393	0.6493	0.6493	0.6493
DCA	0.0510	0.2537	0.6876	0.5413	0.4359	0.4473
ITML	0.5682	0.6365	0.5931	0.6447	0.6445	0.6420
RFD	<b>0.7887</b>	<b>0.8157</b>	<b>0.8367</b>	<b>0.8248</b>	<b>0.8402</b>	0.8745
HFD	<b>0.7788</b>	<b>0.8090</b>	<b>0.8367</b>	0.7258	0.7397	<b>0.9087</b>

low semi-supervised metric learning even in these domains where binary pairwise constraints are no longer possible.

## References

- [1] Aurélien Bellet, Amaury Habrard, and Marc Sebban. A survey on metric learning for feature vectors and structured data. *arXiv preprint arXiv:1306.6709*, 2013.
- [2] Jason V Davis, Brian Kulis, Prateek Jain, Suvrit Sra, and Inderjit S Dhillon. Information-theoretic metric learning. In *ICML*, 2007.
- [3] Chunhua Shen, Junae Kim, Lei Wang, and Anton van den Hengel. Positive semidefinite metric learning with boosting. In *NIPS*, 2009.
- [4] John Blitzer, Kilian Q Weinberger, and Lawrence K Saul. Distance metric learning for large margin nearest neighbor classification. In *NIPS*, 2005.
- [5] Yiming Ying and Peng Li. Distance metric learning with eigenvalue optimization. *The Journal of Machine Learning Research*, 13:1–26, 2012.
- [6] Ratthachat Chatpatanasiri, Teesid Korsrilabutr, Pasakorn Tangchanachaianan, and Boonserm Kijirikul. A new kernelization framework for mahalanobis distance learning algorithms. *Neurocomputing*, 73(10):1570–1579, 2010.
- [7] Sumit Chopra, Raia Hadsell, and Yann LeCun. Learning a similarity metric discriminatively, with application to face verification. In *CVPR*, volume 1, pages 539–546. IEEE, 2005.
- [8] Andrea Frome, Yoram Singer, and Jitendra Malik. Image retrieval and classification using local distance functions. In *NIPS*, volume 2, page 4, 2006.
- [9] Andrea Frome, Yoram Singer, Fei Sha, and Jitendra Malik. Learning globally-consistent local distance functions for shape-based image retrieval and classification. In *Computer Vision, 2007. ICCV 2007. IEEE 11th International Conference on*, pages 1–8. IEEE, 2007.
- [10] Kilian Q Weinberger and Lawrence K Saul. Fast solvers and efficient implementations for distance metric learning. In *ICML*, pages 1160–1167. ACM, 2008.
- [11] De-Chuan Zhan, Ming Li, Yu-Feng Li, and Zhi-Hua Zhou. Learning instance specific distances using metric propagation. In *Proceedings of the 26th Annual International Conference on Machine Learning*, pages 1225–1232. ACM, 2009.
- [12] Dor Kedem, Stephen Tyree, Kilian Weinberger, Fei Sha, and Gert Lanckriet. Non-linear metric learning. In *Advances in Neural Information Processing Systems 25*, pages 2582–2590, 2012.
- [13] Caiming Xiong, David M Johnson, Ran Xu, and Jason J Corso. Random forests for metric learning with implicit pairwise position dependence. In *SIGKDD*, 2012.
- [14] Kiri L Wagstaff, Sugato Basu, and Ian Davidson. When is constrained clustering beneficial, and why? *Ionosphere*, 58(60.1):62–3, 2006.
- [15] Leo Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001.
- [16] David M Johnson, Caiming Xiong, Jing Gao, and Jason J Corso. Comprehensive cross-hierarchy cluster agreement evaluation. In *AAAI Late-Breaking Papers*, 2013.
- [17] Linli Xu, James Neufeld, Bryce Larson, and Dale Schuurmans. Maximum margin clustering. In *NIPS*, pages 1537–1544, 2004.
- [18] Bin Zhao, Fei Wang, and Changshui Zhang. Efficient multiclass maximum margin clustering. In *ICML*, 2008.
- [19] Kai Zhang, Ivor W Tsang, and James T Kwok. Maximum margin clustering made practical. *Neural Networks, IEEE Transactions on*, 20(4):583–596, 2009.

- [20] Minh Hoai and Fernando De la Torre. Maximum margin temporal clustering. In *AISTATS*, 2012.
- [21] Caiming Xiong, David M Johnson, and Jason J Corso. Efficient max-margin metric learning. In *ECDM*, 2012.
- [22] Chunhua Shen, Junae Kim, and Lei Wang. Scalable large-margin mahalanobis distance metric learning. *Neural Networks, IEEE Transactions on*, 21(9):1524–1530, 2010.
- [23] Yang Hu, Jingdong Wang, Nenghai Yu, and Xian-Sheng Hua. Maximum margin clustering with pairwise constraints. In *ICDM*, 2008.
- [24] Hong Zeng and Yiu-ming Cheung. Semi-supervised maximum margin clustering with pairwise constraints. *Knowledge and Data Engineering, IEEE Transactions on*, 24(5):926–939, 2012.
- [25] Alan L Yuille and Anand Rangarajan. The concave-convex procedure. *Neural Computation*, 15(4):915–936, 2003.
- [26] Jon Louis Bentley. Multidimensional binary search trees used for associative searching. *Commun. ACM*, 18(9):509–517, September 1975.
- [27] Marius Muja and David G Lowe. Fast approximate nearest neighbors with automatic algorithm configuration. In *VISAPP (1)*, pages 331–340, 2009.
- [28] Aristides Gionis, Piotr Indyk, Rajeev Motwani, et al. Similarity search in high dimensions via hashing. In *VLDB*, volume 99, pages 518–529, 1999.
- [29] K. Bache and M. Lichman. UCI machine learning repository, 2013.
- [30] Jonathan J. Hull. A database for handwritten text recognition research. *PAMI*, 16(5):550–554, 1994.
- [31] Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images. *Master’s thesis, Department of Computer Science, University of Toronto*, 2009.
- [32] Steven CH Hoi, Wei Liu, Michael R Lyu, and Wei-Ying Ma. Learning distance metrics with contextual constraints for image retrieval. In *CVPR*, 2006.
- [33] Jianbo Shi and Jitendra Malik. Normalized cuts and image segmentation. *PAMI*, 22(8):888–905, 2000.
- [34] Andrew Rosenberg and Julia Hirschberg. V-measure: A conditional entropy-based external cluster evaluation measure. In *EMNLP-CoNLL*, volume 7, pages 410–420, 2007.